Sketching Multidimensional Time Series for Fast Discord Mining

Chin-Chia Michael Yeh, Yan Zheng, Menghai Pan, Huiyuan Chen, Zhongfang Zhuang,
Junpeng Wang, Liang Wang, Wei Zhang, Jeff M. Phillips[†], Eamonn Keogh[‡]
Visa Research, *University of Utah*[†], *University of California, Riverside*[‡]
{miyeh,yazheng,menpan,hchen,zzhuang,junpenwa,liawang,wzhan}@visa.com

***Abstract*—Time series discords are a useful primitive for time series anomaly detection, and the matrix profile is capable of capturing discord effectively. There exist many research efforts to improve the scalability of discord discovery with respect to the *length* of time series. However, there is surprisingly little work focused on reducing the time complexity of matrix profile computation associated with *dimensionality* of a multidimensional time series. In this work, we propose a sketch for discord mining among multi-dimensional time series. After an initial pre-processing of the sketch as fast as reading the data, the discord mining has runtime independent of the dimensionality of the original data. On several real world examples from water treatment and transportation, the proposed algorithm improves the throughput by at least an order of magnitude (50X) and only has minimal impact on the quality of the approximated solution. Additionally, the proposed method can handle the dynamic addition or deletion of dimensions inconsequential overhead. This allows a data analyst to consider "what-if" scenarios in real time while exploring the data.**

***Index Terms*—multidimensional time series, discord mining, similarity join**

I. Introduction

Time series discords are a simple, effective, and robust primitive for detecting anomalies in time series data [1–3]. While there have been dozens of algorithms proposed to compute discords in the last twenty years, in recent years the *matrix profile* (MP) has emerged as the most effective and versatile computation tool for discovering discords [4, 5]. There have been numerous efforts on improving the scalability of the MP. For example, Zimmerman et al. [6] improve the computational speed by exploiting hardware, Zhu et al. [7] introduced an anytime algorithm with a fast convergence rate, and efficient approximation of the MP is proposed in [8]. However, all these ideas only consider univariate time series. In other words, these works focus on reducing the time complexity with respect to the *length* of the input time series, while the runtime cost that is associated with *dimensionality* of multidimensional time series is unchecked. The word "*dimensionality*" is used inconsistently in the time series literature. To be clear, in this work *dimensionality* means the number of individual and concurrent streams, for example, a three-dimensional medical time series might contain {`ECG|respiration|temperature`}.

The ability to scale with respect to dimensionality is crucial in many domains. For instance, there are hundreds of millions of merchants in modern-day financial transaction networks. If a regulatory agency wants to monitor the activity of each merchant for detecting suspicious behavior in real-time, it will need to maintain hundreds of millions of MPs for discord discovery. In this paper, we propose a sketching algorithm on multidimensional time series for discord mining so that it can dramatically reduce the cost associated with monitoring multiple dimensions simultaneously.

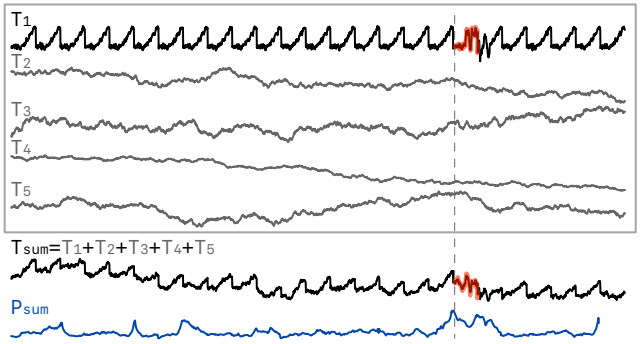

Fig. 1: In the multidimensional time series $\mathbf{T} = [T_1, ..., T_5]$, the time series discord occurs in T_1 can be discovered even if the matrix profile P_{sum} is computed using the summed time series $T_{sum} = T_1 + \cdots + T_5$.

To preview our ideas, consider the example shown in Fig. 1 which demonstrates how simple addition-based dimension reduction works for multidimensional time series discord mining. The multidimensional time series $\mathbf{T}$ has five dimensions $[T_1, ..., T_5]$, and the time series discord occurs in T_1. We can add all the dimensions together (i.e., T_{sum}) and compute the associated MP (i.e., P_{sum}). Due to the robustness of the MP method, P_{sum} still reveals the temporal location of the discord that occurred only in T_1 despite the existence of the other irrelevant dimensions. Because we only need to compute one MP instead of five MPs, a 5X improvement in speed is achieved. Before continuing, we wish to ward off a potential misunderstanding. Clearly, it is not meaningful to add dollars and yen, much less dollars and temperature. However, we *can* meaningfully add z-normalized time series of arbitrary origin. The z-normalization step converts the data into unitless shapes.

As pointed out in [9], if more and more irrelevant dimensions are included, eventually the discord in T_1 will be missed if only the MP of the summed time series is examined. To solve this problem, we designed a sketch [10–12] where MP can be built on the sketch of the multidimensional time series

and discord property can be preserved. The contributions of this paper include:

- This is the first work to apply sketching techniques to multidimensional time series discord mining with theoretical guarantees.
- After computing the sketch in time as fast as reading the data, we guarantee to find time points of significant discords with high probability efficiently and in time independent of the number of dimensions.
- Experiments demonstrate that with an insignificant drop in accuracy, the sketched discord mining can achieve a 50X speed-up. This utility is realized on several real-world data sets from water treatment, distributed systems, and transportation.

II. Background

A. Related Work

Our proposed ideas have an interesting historic context, in that they resemble *Dorfman* testing (*group* testing, *pooled* testing, etc.) [13]. This is a procedure used to reduce the cost of screening a large number of individuals for infectious diseases. It works by compositing a set of individual specimens (e.g., blood or urine) into a common pool. If the pool tests negative, all individuals within it are diagnosed as negative. If the pool tests positive, retesting is needed to find the positive individual(s). The reader will appreciate that in our example in Fig. 1, we had a "pool" of five time series, and the summed time series did indeed "test positive" for a discord.

How to efficiently pool data into a compressed representation while provably preserving critical information has taken on the name of *sketching*. These ideas have proven invaluable in domains such as network monitoring [10, 14], linear algebra [15, 16], machine learning [11, 17, 18], and spatial statistics [19]. To the best of our knowledge, this is the first time that *Dorfman testing* or *sketching* has been used in time series analysis.

In recent years, there has been an increasing focus on multidimensional time series [9, 20–24]. For example, Yeh et al. [9] adopted the matrix profile [4] idea for mining motifs in multidimensional time series. Gao et al. [22] developed a variable-length subdimensional motif discovery system for multidimensional time series. Similar to our work, the system introduced in [20] also used a type of random projection as a subroutine. However, instead of using it for mining time series discords, the proposed system in [20] is a subdimensional motif discovery system similar to [22]. Other tasks, such as time series forecasting [23, 24] and anomaly detection [21] have also been studied for multidimensional time series. Among these works, research efforts on time series anomaly detection are more relevant to our work because time series discords not only can be utilized to solve the anomaly detection problem but many independent papers have shown they are very competitive compared to the state-of-the-art [3, 25–27]. However, none of these prior works provide a scalable discord mining algorithm for multidimensional time series.

The matrix profile is an efficient way to discover discords in time series [4]. In recent years, there have been many attempts to further improve the computational speed of the matrix profile algorithm through approximation [7, 8, 28]. Nevertheless, all of the aforementioned works focused on reducing the computational cost associated with the *length* of the input time series rather than the *dimensionality* of the time series. These works cannot be used to resolve the scalability problem raised in this paper. Dimensionality reduction methods such as PAA [29], DWT [30], and DFT [31] are widely used in time series data mining to reduce run time [32]. However, as we noted earlier, here dimensionality refers to the *length* of the time series, not the number of time series; therefore, they also do not address the scalability problem associated with dimensionality.

B. Definitions and Notation

We begin by defining the data type of interest, *time series* and *multidimensional time series*:

Definition 1. A *time series* $T \in \mathbb{R}^n$ is an array of real valued numbers $t_i \in \mathbb{R} : T = [t_1, t_2, ..., t_n]$ where n is the length of T. A *multidimensional time series* $\mathbf{T} \in \mathbb{R}^{d \times n}$ is a collection of single dimensional time series $\mathbf{T} = \left[T^{(1)}, T^{(2)}, ..., T^{(d)}\right]$ where d is the dimensionality of $\mathbf{T}$.

Since time series discords are a *local* property of a time series, we are not interested in the *global* properties of a time series, but the *local subsequences* [5]:

Definition 2. A *subsequence* $T_{i,m} \in \mathbb{R}^m$ of a time series T is a length m contiguous subarray of T starting from position i. Formally, $T_{i,m} = [t_i, t_{i+1}, ..., t_{i+m-1}]$. Note, we use $T_{i,m}^{(j)}$ to denote the ith subsequence in jth dimension of a multidimensional time series $\mathbf{T}$.

Because time series discords concern the nearest neighbor (*1NN*) relation between a query subsequence and all subsequences of a given time series, we define a *1NN_dist* function which computes the nearest neighbor distance between the query subsequence with a given time series.

Definition 3. Given a query subsequence $T_{q,m} \in \mathbb{R}^m$ and a time series $T \in \mathbb{R}^n$, a *1NN distance function 1NN_dist*$(T_{q,m}, T)$ computes and returns the distance between $T_{q,m}$ and the nearest neighbor subsequence in T, i.e., $\min_{T_{i,m} \in T}$ *dist*$(T_{q,m}, T_{i,m})$ where *dist*$(\cdot, \cdot)$ computes the z-normalized Euclidean distance between the two inputs.

We use z-normalized Euclidean distance function in this work. Note, *1NN_dist*$(\cdot, \cdot)$ and *dist*$(\cdot, \cdot)$ look similar, but they are very different functions. *1NN_dist*$()$ computes the one nearest neighbor distance between a subsequence and a time series and *dist*$()$ computes the distance (e.g., z-normalized Euclidean distance) between two subsequences. The particular local properties that we seek to capture are *time series discords*:

Definition 4. Given a training time series T_{train}, a testing time series T_{test} and a subsequence length m, the *time series*

discord of T_{test} is the subsequence of length m in T_{test} with largest nearest neighbor distance to subsequences in T_{train}, i.e., $\arg\max_{T_{i,m} \in T_{\text{test}}} \mathit{1NN\_dist}(T_{i,m}, T_{\text{train}})$. When detecting the discord within a single time series, i.e. $T_{\text{test}} = T_{\text{train}}$, the definition is also applied.

Additionally, since we are considering multidimensional time series, we extend the aforementioned single-dimensional time series discord to the following:

Definition 5. Given a multidimensional training time series $\mathbf{T}_{\text{train}} = \left[T^{(1)}_{\text{train}}, T^{(2)}_{\text{train}}, ..., T^{(d)}_{\text{train}}\right]$, a multidimensional testing time series $\mathbf{T}_{\text{test}} = \left[T^{(1)}_{\text{test}}, T^{(2)}_{\text{test}}, ..., T^{(d)}_{\text{test}}\right]$ and a subsequence length m, the *time series discord* of $\mathbf{T}_{\text{test}}$ is the subsequence of length m in $\mathbf{T}_{\text{test}}$ with the largest nearest neighbor distance (where the nearest neighbors are in $\mathbf{T}_{\text{train}}$ of the same dimension as the subsequence). Formally:

$$T^{(j^*)}_{i^*,m} = \arg\max_{\substack{T^{(j)}_{i,m} \in T^{(j)}_{\text{test}} \\ T^{(j)}_{\text{test}} \in \mathbf{T}_{\text{test}}}} \mathit{1NN\_dist}(T^{(j)}_{i,m}, T^{(j)}_{\text{train}})$$

The basic time series discord only needs to return the time index i^* at the start of the subsequence, $T^{(j^*)}_{i^*,m}$. The *dimensional time series discord* identifies both the time index i^*, and also the dimension j^* in which this most anomalous pattern occurs.

The most efficient method of locating the single dimension time series discord exactly is the *matrix profile* [4, 5].

Definition 6. A ab-join matrix profile [5] $P \in \mathbb{R}^{n_{\text{test}}-m+1}$ of a time series $T_{\text{test}} \in \mathbb{R}^{n_{\text{test}}}$ and a time series $T_{\text{train}} \in \mathbb{R}^{n_{\text{train}}}$ is a meta time series annotating T_{test} that stores the distance between each subsequence of length m in T_{test} and its nearest neighbor in T_{train}. When $T_{\text{test}} = T_{\text{train}}$, it is the definition of *self-join* matrix profile [5].

The matrix profile P of a time series T ($|T| = n$) is shown in Fig. 2. The length of P is $n - m + 1$ where m is the subsequence length. By examining the location of the largest value in P, the embedded time series discord can be discovered.

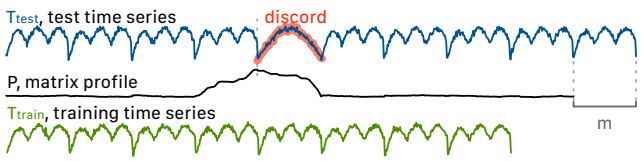

Fig. 2: The matrix profile P of the time series T_{test} is obtained by joining it with another time series T_{train}. The highest point on P correspond to the locations of discord (red).

The matrix profile can be computed exactly in $O(n_{\text{train}} n_{\text{test}})$ time complexity. Since the multidimensional time series discord is defined as the single dimension discord with largest nearest neighbor distance (Def. 5), the multidimensional time series discord can be discovered exactly by running matrix profile d times for a d dimensional time series. In other words, the time complexity for this basic solution is $O(d \cdot n_{\text{train}} n_{\text{test}})$, and the goal of this work is to further reduce this complexity.

III. Algorithm

The proposed method has two stages: 1) the sketching stage and 2) the detection stage. In the sketching stage, each dimension is randomly assigned to different groups. With the group assignment initialized, the sketched time series are used in the second stage for fast detection of time series discords. In addition to introducing the two stages of the proposed method, we also discuss how the proposed method handles the addition/deletion of dimensions.

A. Sketching

The sketching method is based on a count sketch [10]. It maps from d time series (one for each dimension) to k time series, one for each group. The count sketch uses two pairwise independent hash functions $h \in \mathcal{H}$ with $h : [d] \to [k]$ and $s \in \mathcal{S}$ with $s : [d] \to \{-1, +1\}$. We use $[d]$ as a short hand for $\{0, 1, \ldots, d-1\}$. The first hash function h maps each dimension to a group, the second one endows each dimension with a sign. Recall that the hash functions h and s are deterministic, but their selection from the families $\mathcal{H}$ and $\mathcal{S}$ is random.

For a multidimensional time series $\mathbf{T} = \left[T^{(1)}, T^{(2)}, ..., T^{(d)}\right]$ and each group index $g \in [k]$, let $J_g = \{j \in [d] \mid h(j) = g\}$ be the set of dimensions mapped to group g. Then these encode an aggregated time series $\mathbf{R}^{(g)} = \sum_{j \in J_g} s(j) T^{(j)}$.

As is common in sketching, the algorithm (see Alg. 1) is simple, and in this case takes $O(nd)$ time, basically just reading the input data. This can be done to create $\mathbf{R}_{\text{train}} \leftarrow \textsc{Sketch}(\bar{\mathbf{T}}_{\text{train}}, k)$ and $\mathbf{R}_{\text{test}} \leftarrow \textsc{Sketch}(\bar{\mathbf{T}}_{\text{test}}, k)$ using the same hashing functions, where $\bar{\mathbf{T}}_{\text{train}}$ and $\bar{\mathbf{T}}_{\text{test}}$ are the z-normalized multidimensional time series of the train and test data, respectively.

Algorithm 1 Sketching Algorithm

Input: multidimensional time series $\mathbf{T} \in \mathbb{R}^{d \times n}$, sketch dimension k
Output: sketched matrix $\mathbf{R} \in \mathbb{R}^{k \times n}$
function $\textsc{Sketch}(\mathbf{T}, k)$
  $\mathbf{R} \leftarrow$ zero matrix with size $k \times n$
  for $j \in [0, \cdots, d-1]$ **do**
    $i = h(j)$
    $\mathbf{R}^{(i)} = \mathbf{R}^{(i)} + s(j) T^{(j)}$
  Return $\mathbf{R}$

B. Detection

Given the sketch matrices $\mathbf{R}_{\text{train}}$ and $\mathbf{R}_{\text{test}}$ for the training and testing multidimensional time series, and the subsequence length m, we can use Alg. 2 to find the top-1 time series discord. The function $\textsc{AbJoinMP}(\cdot)$ returns both the index and the score for the discord.

The discord detection algorithm runs in two phases. In the first phase to detect the time of the discord, Alg. 2, it only considers the sketched time series $\mathbf{R}_{\text{train}}$ and $\mathbf{R}_{\text{test}}$, and identifies the time of the discord subsequence in the test data

Algorithm 2 Discord Time Detection Algorithm

Input: sketched training time series $\mathbf{R}_{\text{train}}$, testing time series $\mathbf{R}_{\text{test}}$, subsequence length m
Output: discord time index i^*, discord group g^*,
1 **function** TIME-DETECTION($\mathbf{R}_{\text{train}}$, $\mathbf{R}_{\text{test}}$, m)
2 $\hat{s}_{\text{bsf}} \leftarrow 0$
3 **for** $g \in [0, \cdots, k-1]$ **do**
4 $i, s^{(g)} \leftarrow \text{ABJOINMP}(R^{(g)}_{\text{train}}, R^{(g)}_{\text{test}}, m)$ ▷ Def. 6
5 **if** $s^{(g)} > \hat{s}_{\text{bsf}}$ **then**
6 $g_{\text{bsf}} \leftarrow g$, $\hat{s}_{\text{bsf}} \leftarrow \hat{s}^{(g)}$, $i_{\text{bsf}} \leftarrow i$
7 **return** $i^* \leftarrow i_{\text{bsf}}$, $g^* \leftarrow g_{\text{bsf}}$

Algorithm 3 Discord Dimension Detection Algorithm

Input: training time series $\mathbf{T}_{\text{train}}$, testing time series subsequences $\mathbf{T}_m$, subsequence length m, group of time series J_g
Output: discord dimension index j^*
1 **function** DIMENSION-DETECTION($\mathbf{T}_{\text{train}}$, $\mathbf{T}_m$, m, J_{g^*})
2 $s_{\text{bsf}} \leftarrow 0$
3 **for** $j \in J_{g^*}$ **do**
4 $\_, s^{(j)} \leftarrow \text{ABJOINMP}(T^{(j)}_{\text{train}}, T^{(j)}_m, m)$ ▷ Def. 6
5 **if** $s^{(j)} > s_{\text{bsf}}$ **then**
6 $j^* \leftarrow j$, $s_{\text{bsf}} \leftarrow s^{(j)}$
7 **return** j^*

i^*, and the group g^* of dimensions which may contain the discord subsequence. To do this, it treats the sketched time series as just k-dimensional time series, and checks each of the k possible sketched-to dimensions. Since this phase only operates on the sketched data, its runtime is independent of the number of dimensions d.

In the second phase showed in Alg. 3, it recovers the dimension which leads to the discord. It takes in the test data subsequences $\mathbf{T}_m$ of multidimensional time series, starting at index i for a length of m that contains all dimensions from the test set for that time window; it actually only needs those for indexes $j \in J_{g^*}$ in group g^*. Then for this fixed time window in the test data, it checks each of the dimensions that contributed to group g^* to see which one is the discord using standard Matrix Profile ab-join. Note that users have the option to refine the current approximated solution by performing an additional Matrix Profile join operation on the entire sequence of the identified dimension to find a subsequence with an even higher discord score. This functionality has been implemented in the released source code.

We use the matrix profile [5] method to locate and score the time series discord within a given pair of training/testing single-dimension time series for the TIME-DETECTION algorithm. In particular, we use the SCAMP algorithm [6] to compute the matrix profile. Because SCAMP is not the only matrix profile algorithm, it is possible to replace it with a faster and approximated algorithm from [7, 8, 28].

The time complexity of the discord time detection is $O(k \cdot n_{\text{train}} n_{\text{test}})$, where the $n_{\text{train}} n_{\text{test}}$ part of the big-O notation is from the matrix profile algorithm (see Def. 6). To detect the dimension of the discord, another $O((d/k) n_{\text{train}})$ time is needed. The exact multidimensional discord mining algorithm, without the $O(d(n_{\text{train}}+n_{\text{test}}))$ sketching step is $O(d \cdot n_{\text{train}} n_{\text{test}})$.

To extend the proposed method to the single time series scenario, the user can input the same multidimensional time series to both $\mathbf{T}_{\text{train}}$ and $\mathbf{T}_{\text{test}}$; then use the self-join variate of the matrix profile algorithm instead of the ab-join variant in Alg. 2. It is also possible to extend the method to streamline time series. To achieve such extension, lines 4-5 in Alg. 1 need to be implemented to only invoke whenever a new test data is received. The matrix profile algorithm used in Alg. 2 will also need to be replaced with a streaming variant such as the ones presented in [5, 8, 28].

C. Addition/Deletion of Dimensions

The proposed method can also be extended to handle the addition or deletion of dimensions. Since the count sketch is "linear" it can be updated, via additions, deletions, and modifications. Such scenarios could happen when sensors are added or removed from a system. For instance, in the case where dimension $j \in [d]$ is deleted, the gth sketched time series where $g = h(j)$ is updated as $\mathbf{R}^{(g)} = \mathbf{R}^{(g)} - s(j)\mathbf{T}^{(j)}$. If only the ith time point of the jth dimension is updated, to increase its value by a value δ, then we can update $R^{(g)}_i = R^{(g)}_i + s(j)\delta$. The detection algorithms are unchanged.

As a technical issue, it may not be appropriate to add a new time series dimension that is only available for part of the training time. At the beginning of the membership, it could create an artificial "step" shape. While we may be able to mark this start point as to be avoided in the Discord Time Detection algorithm, we mostly recommend avoiding this complication, and suggest that only time series that starts at the same time should be added together.

D. Analysis of Accuracy of Algorithm

We analyze the accuracy of the algorithm by extending the analysis of the CountSketch [10]. In particular, we analyze the probability that a discord subsequence of length m is detected as the discord in the sketched representation. The details of the analysis are in Appendix. We show several results.

First, if a time series $T^{(j)}$ is randomly placed in group g, and multiplied by sign $s(j) \in \{-1, +1\}$ in the sketched time series $\mathbf{R}^{(g)}$, then for any point i the expected value $E[s(j) \cdot R^{(g)}_i] = T^{(j)}_i$ (see Lemma 1 in Appendix). This expectation is only over randomness in the choice of $h \in \mathcal{H}$ and $s \in \mathcal{S}$.

Next we consider how large a discord score of time series dimension $T^{(j)}$ needs to be to show up as the discord in the sketched time series $\mathbf{R}^{(g)}$ We provide a fairly crude bound that does not depend on the randomness of the data, only using that the dimensions are each z-normalized. Set $k = \sqrt{d}$ and consider a discord $T^{(j)}_{i,m}$ that has distance of $\|\Delta\| = dist(T^{(j)}_{i,m}, T^{(j)}_{i',m})$ where $T^{(j)}_{i',m}$ is the nearest neighbor of $T^{(j)}_{i,m}$ in the training set. Then if $\|\Delta\| > \tau = \frac{1}{\sqrt{\delta}} m d^{1/4}$, then it will be detected as a discord in $\mathbf{R}^{(g)}$ with probability at least $1-\delta$.

The above bound requires τ grows with $d^{1/4}$ and generally must be much larger than 1 standard deviation in data value, so we analyze a stronger bound if we assume some noisy periodic structure in each time series. Under the *η-periodic assumption*, we assume each time series basically has a repeated structure

where each element deviates from that sequence with standard deviation at most η (after z-normalizing). Under this setting, with high probability (the probability of failure decreases exponentially as n_{train} grows), if $\|\Delta\| > \tau > 2m\eta$, then it will be detected as the discord in $\mathbf{R}^{(g)}$ (see Lemma 2 in Appendix).

IV. Experiments

We first employ synthetic data to assess the quality of the approximated solution. Next, we provide case studies with a public transportation dataset and a financial transactions dataset to demonstrate the broad applicability of the proposed method. Finally, we utilize a water treatment/distribution system dataset to evaluate the effectiveness of the proposed algorithm in the anomaly detection task. *It is important to note that anomalies and discord are distinct concepts.* Anomalies are defined within the specific context and semantics of the dataset, often annotated manually. Conversely, discords are domain-independent, can be used to identify potential anomalies, and are explicitly defined using Def. 5. The source code can be found in [33].

A. Synthetic Data

In this section, we use synthetic data to evaluate the quality of the approximated solution compared to the exact solution. The quality of the solution is measured by how well the proposed algorithm solves the equation from Def. 5.

Experiment Setup: We generate synthetic datasets by creating random walk time series. It is important to note that random walk time series pose a significant challenge for time series discord mining, as the discord does not exhibit a visually distinct pattern compared to the rest of the time series. The length of the generated time series is fixed at 10,000, and we vary the dimensionality d of the time series. We specifically focus on exploring different values of d since it is the primary challenge addressed by our proposed method. We set d to the following values: 250, 500, 1,000, 2,500, 5,000, 7,500, and 10,000. During the experiment, we use a subsequence length of 100. The experiment is repeated 100 times. We set hyperparameter k to $\lceil\sqrt{d}\rceil$ to optimize the $O(k + d/k)$ term from combining phases of detection Algs. 2 and 3.

Performance Measurement: The objective of this section is to evaluate the accuracy of the approximated solution in comparison to the exact solution. To begin with, we calculate the discord score, which represents the nearest neighbor distance, for each subsequence across all dimensions. Subsequently, we generate a ranked list by assigning ranks to all the subsequences based on their discord scores. To measure the quality of the approximated solution, we compute the success rate over 100 trials. We consider the approximated algorithm successful when the identified discord is ranked within the top 0.01% of the list.

Result: The results of the experiments are illustrated in Fig. 3. The speedup presented takes into account the total runtime, which includes both the sketching and detection processes.

The proposed method demonstrates a noticeable improvement in terms of speedup as we increase the value of d. At the highest tested value of d (i.e., 10,000), the speedup reaches 50X with an almost 100% success rate.

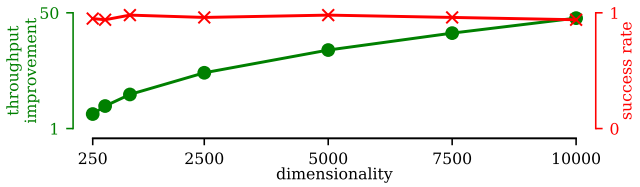

Fig. 3: The proposed method improves throughput by 50X while maintaining almost perfect success rate when d = 10000.

To gain further insights into the quality of the approximate solution, we analyze the distribution of discord scores (i.e., the nearest neighbor distance) for three sets of subsequences. The first set encompasses all subsequences, representing the overall distribution of discord scores. The second set comprises the true discords discovered using the exact algorithm, while the last set consists of the discords identified using the proposed sketching method. Fig. 4 displays the density plots illustrating the distribution associated with each set of subsequences. The d is set to 1,000 in this figure. Notably, the distribution of discords obtained through the proposed approximate method is distinctly different compared to the distribution of discord scores for all subsequences. The distribution of the approximated discords is much more similar to the distribution of discords obtained through the exact method. It is important to note that the task of discord mining in the random walk dataset presents a greater level of difficulty compared to other datasets. This increased challenge arises from the fact that the discord, representing the most unusual subsequence (red distribution), is not as easily distinguishable from the other subsequences. In the real datasets (see Section IV-D and Section IV-C), the discord scores between the exact discord and the sketched ones are much smaller. In Fig.6, the discord score of the subsequence discovered by the exact algorithm deviates from the mean of the subsequences discovered by the approximated algorithm by 1.97 standard deviations. Similarly, in Fig.8, the discord score of the subsequence discovered by the exact algorithm deviates from the mean by 2.11 standard deviations.

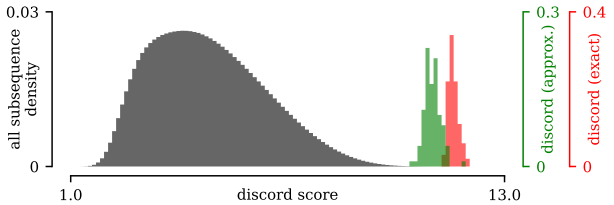

Fig. 4: The density distribution of discord scores for all subsequences (gray), the discords found using the proposed approximation method (green), and the discords found using the exact method (red).

B. Taipei Mass Rapid Transit (MRT)

The Taipei MRT system is a metro system that moves over two million people daily in Taipei and its surrounding satellite towns. It could be actionable to bring the operator's attention to the unusual patterns in the ridership data. For example, a traffic manager may spot an anomaly, perhaps a sudden exodus from a station caused by customers fleeing a crime [34], and she may be able to intervene stopping trains before they arrive at that station. In this section, we apply the proposed algorithm to discover multidimensional time series discord from the per-hour ridership data. Discords typically correspond to interesting and unique events.

Dataset: The original dataset is downloaded from [35], and we use the version organized by the authors of [36]. The dataset consists of time series measuring the number of people entering and exiting each station per hour from November 1, 2015 to March 31, 2017. There are 108 stations in the dataset.

Experiment Setup: We focus on the time series corresponding to the number of people entering and exiting each of the 108 stations. We set the subsequence length m to two days (i.e., 48) and the sketching parameters to $k = \lceil \sqrt{d} \rceil$ when applying the proposed method. A list of discovered discords is returned by the algorithm order based on the nearest neighbor distance (i.e., discord score) from large to small. We ignore citywide incidents like typhoon days when inspecting the returned discords.

Result: The discord with the largest discord score is shown in Fig. 5, and it is discovered from *Nangang Software Park Station*.

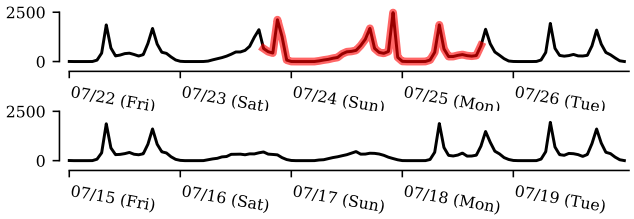

Fig. 5: The time series is from Nangang Software Park station in 2017. The top figure shows the discovered time series discord (red) with additional context. The y-axis is the number of people entering and exiting the station per hour.

In the top figure, we show the discovered discord with extra dates preceding and following the discord for context. Because the area around Nangang Software Park Station consists mostly of office buildings, it has morning and afternoon peaks during workdays. The afternoon peaks that happen during the weekend are very unusual for the station. As shown in Fig. 5.*bottom*, the time series from the previous week has no peak during the weekends. In other words, the discovered discord indeed captures some special events that happen around the station. After inspecting the area surrounding the station on a street map, there is a parking lot called *Nangang C3 Field* [37] that can also be used as a concert ground for 40,000 attendances. Upon inspecting the event schedule for the venue [37], there were indeed concerts held on both 7/23 and 7/24 by a popular rock band Mayday [38]. The rock concert explains the peaks/discord discovered on 7/23 and 7/24.

In addition to the exploratory analysis, we compare the distribution densities using a methodology similar to that described in Section IV-A. The resulting figure is presented in Fig. 6. It is evident that the discovered discords are clearly outliers when compared to the distribution of all subsequences. Furthermore, the proposed method achieved an average speedup of 3.6X compared to the exact solution. Please note that in this particular case, there is only one dataset, and as a result, there is also only one true discord indicated by a single line.

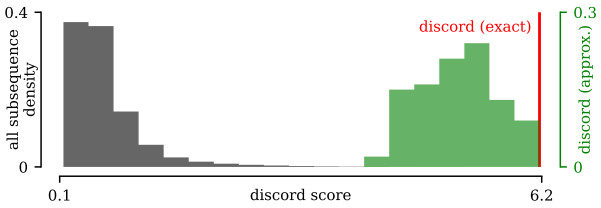

Fig. 6: The density distribution of discord scores for all subsequences (gray) and the discords found using the approximation method (green). The discord score of the discord found using the exact method is indicated by a red line.

C. Payment Network.

There are millions of transactions among different entities processed daily on a modern payment network. It is important to monitor the activity of these entities for unusual patterns (i.e., discords) as such events could have undesirable effects on the software or hardware infrastructures built around the payment network. For example, a sudden fluctuation in the transaction volume could lead to undesirable declines if the fluctuation is not attended to. To demonstrate the utility of the proposed method, we apply our discord mining algorithm to the per-hour transaction volume multidimensional time series where different dimensions are different categories of merchants. Similar to Section IV-B, multiple discords are returned by our algorithm and we order them based on the "discordness" of the returned pattern (i.e., the distance with the pattern's nearest neighbor).

Dataset: We prepare the multidimensional time series from our internal transaction database. The per category per hour time series is aggregated using transaction data from January 1st, 2018 to July 1st, 2021. The length of the time series is 30,600, and the number of categories is around 1,000.

Experiment Setup: When applying the proposed method, we set the subsequence length m to 36 or 1.5 days. The sketching parameter is set to $k = \lceil \sqrt{d} \rceil$. Because it is beneficial to examine the second/third discord in the time series, our algorithm once again returns multiple discords as we did in Section IV-B. The discords are ordered based on the nearest neighbor distance (i.e., discord score) from large to small.

Result: The top three discovered discords are shown in Fig. 7.

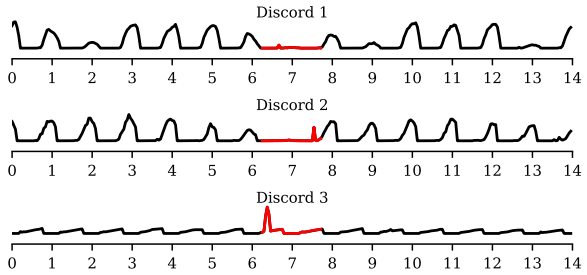

Fig. 7: The top three discovered time series discords. The discords (red) are plotted with additional context, and total of 14 days (336 hours) of time series are plotted.

The first discovered discord (i.e., Discord 1) consists of a period of time with relatively small numbers of transactions with a small bump around the 7th-day mark. The second discovered discord (i.e., Discord 2) also captures a period of time where the number of transactions is relatively low. However, there is a sharp spike near the middle of the 7th day. One possible reason for the low transaction volume could be that majority of merchants from the corresponding categories are closed during that time. The small bump around the 7th-day mark in Discord 1 indicates there might be a subset of merchants still doing business during that time. For Discord 2, the sharp spike that happens right before the normal business volume around the 8th-day mark could mean that merchants of the Discard 2 category could still be accepting orders, but push the payment process to the beginning of the next business day. The third discovered discord (i.e., Discord 3) could indicate the occurrence of a special sales event. Time series discords are capable of capturing both high and low-volume events that occur in the payment network. The speedup of the proposed method in throughput relative to the exact solution is 13 for this dataset.

We further compared the distribution densities similar to the experiment conducted with the MRT data. The resulting figure is displayed in Fig. 8, showcasing that the discovered discords are distinctly outliers in comparison to the distribution of all subsequences. The discord identified by the approximate method exhibits a significantly higher similarity to the exact discord compared to the synthetic dataset (Figure 4), which demonstrates the effectiveness of the sketching algorithm on the real-world dataset.

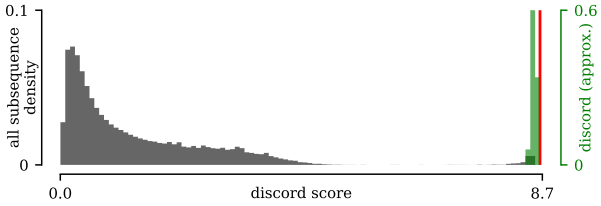

Fig. 8: The density distribution of discord scores for all subsequences (gray) and the discords found using the approximation method (green). The discord score of the discord found using the exact method is indicated by a red line.

D. Water Treatment and Distribution

In order to showcase the efficacy of the approximate sketching algorithm in anomaly detection, we employ a dataset obtained from a water treatment and distribution system. Our primary objective is to conduct a quantitative comparison between the proposed discord mining algorithm and other existing methods commonly used for anomaly detection.

Dataset: We performed experiments with SWaT [**?**] and WADI datasets [39] in detecting attacks on cyber physical systems. SWaT dataset contains the measurement of a scaled-down version of a real-world industrial water treatment plant under the normal and attacked scenarios [**?**]. A total of 51 sensors and actuators are measured for 11 days with a 1 Hz sampling rate. The first 7 days consist of the measurement from normal operation. The system is attacked 41 times during the last 4 days. WADI dataset consists of measurements from a water distribution system with 123 sensors and actuators [39]. The measurement takes place in 16 days where the system operates normally in the first 14 days and is attacked 14 times in the last 2 days. The data sampling rate is 1 Hz.

Experiment Setup: We compare the proposed approximated multidimensional discord mining matrix profile (Discord/Fast) to methods like exact multidimensional discord mining matrix profile (Discord/Exact), 1NN [40], LOF [40, 41], OC-SVM [40, 42, 43], and MAD-GAN [21]. MAD-GAN is one of the state-of-the-arts for the datasets, and the other methods are classic anomaly detection algorithms [21].

We follow the following procedure to apply discord-based methods (Discord/Fast and Discord/Exact) to the anomaly detection problem. First, we identify the dimension that contains the multidimensional discord using either the exact multidimensional discord mining algorithm or the proposed sketching algorithm. Let us assume that the discord is found in dimension j. In the anomaly detection test dataset, we have the label of each subsequence, so the next step is to compute the anomaly score associated with each subsequence. We extract the jth dimension of the test time series $T^{(j)}_{\text{test}}$ and join it with $T^{(j)}_{\text{train}}$ using the matrix profile algorithm. In other words, for each subsequence in $T^{(j)}_{\text{test}}$, we find its nearest neighbor in $T^{(j)}_{\text{train}}$ and compute the distance between each pair of nearest neighbors. We use these nearest neighbor distances as the anomaly score for each subsequence in $T^{(j)}_{\text{test}}$. We set $k = \lceil \sqrt{d} \rceil$ when using the Discord/Fast method.

Performance Measurement: With the label and anomaly score of each subsequence in $T^{(j)}_{\text{test}}$, the performance measurements adopted for both datasets are ROC-AUC (AUC) scores. We choose to use AUC score instead of the more common F1 score for these datasets because the anomaly scores need to be converted to binary predictions for evaluating the F1 score of a method. As pointed out by Kim et al. [**?**], converting the anomaly scores to binary labels for time series anomaly detection is itself a challenging problem. Since our paper is focusing on mining time series discord, we use AUC score to avoid the additional complication associated with the anomaly score conversion.

TABLE I: The proposed method achieves comparable performance to the best alternative with reduced runtime.

	SWaT		WADI	
Method	AUC	Time (sec.)	AUC	Time (sec.)
1NN	0.82	8,534	0.47	5,057
LOF	0.79	18,778	0.47	30,241
OC-SVM	0.82	42,934	0.52	130,221
MAD-GAN	0.81	3,273	0.45	3,627
Discord/Exact	**0.83**	2,488	0.62	2,544
Discord/Fast	0.76	**821**	**0.72**	**653**

Result: The AUC scores for all the tested methods on both datasets are summarized in Table I. For the SWaT dataset, all methods achieve comparable AUC scores. Since the proposed method also has similar performance in terms of AUC, its advantage in running time makes it more desirable than the others. The performance difference between different methods on the WADI dataset is even greater. The discord-based methods are noticeably better than the alternatives. Note that the dimension located by the exact discord-based method could sometimes be inferior compared to the one found by approximated method because accurate discord mining does not necessarily translate to accurate anomaly detection.

To evaluate the robustness of different methods, we add 200 random walk time series to each dataset and redo the experiments in Table II. The expanded SWaT dataset consists of 251 and the expanded WADI consists of 323 dimensions.

TABLE II: The proposed method is robust against added random walk dimensions.

	SWaT		WADI	
Method	AUC	Time (sec.)	AUC	Time (sec.)
1NN	0.66	9,857	0.47	6,010
LOF	0.76	28,282	0.42	35,038
OC-SVM	0.79	186,623	0.47	368,884
MAD-GAN	0.49	7,814	0.46	5,318
Discord/Exact	**0.83**	12,755	0.62	7,731
Discord/Fast	0.76	**2,261**	**0.64**	**1,382**

When contrasting the performances in Table II with the ones in Table I, the SWaT dataset's AUC degrades more compared to the WADI dataset. Out of all the tested methods, LOF and the proposed method are more robust against the added noise. For the discord-based methods, we found that the discovered discords are from one of the original dimensions rather than the added noisy dimensions. The proposed discord-based methods better ignore the noisy dimensions when detecting an anomaly. The speedup of the approximated method compared to the exact method is around 5.6X for both datasets.

V. Conclusion

The paper proposed an efficient sketching-based multidimensional discord mining algorithm to quickly identify the discord. The algorithm improves the throughput by almost 50x and provably maintains high accuracy compared to the exact discord mining algorithm. Experiments are performed with four datasets from various domains to demonstrate the effectiveness and utility of the proposed algorithm. For future work, we consider extending the sketching-based method for subsequence classification [44] or to explore the data privacy issues associated with time series [45].

Acknowledgement

Jeff M. Phillips thanks NSF III-1816149.

References

[1] E. Keogh, J. Lin, and A. Fu, "Hot sax: Efficiently finding the most unusual time series subsequence," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*. Ieee, 2005, pp. 8–pp.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[3] R. Wu and E. Keogh, "Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[4] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, "Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets," in *2016 IEEE 16th international conference on data mining (ICDM)*. Ieee, 2016, pp. 1317–1322.

[5] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, Z. Zimmerman, D. F. Silva, A. Mueen, and E. Keogh, "Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile," *Data Mining and Knowledge Discovery*, vol. 32, pp. 83–123, 2018.

[6] Z. Zimmerman, K. Kamgar, N. S. Senobari, B. Crites, G. Funning, P. Brisk, and E. Keogh, "Matrix profile XIV: scaling time series motif discovery with gpus to break a quintillion pairwise comparisons a day and beyond," in *Proceedings of the ACM Symposium on Cloud Computing*, 2019, pp. 74–86.

[7] Y. Zhu, C.-C. M. Yeh, Z. Zimmerman, K. Kamgar, and E. Keogh, "Matrix profile XI: SCRIMP++: time series motif discovery at interactive speeds," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 837–846.

[8] Z. Zimmerman, N. S. Senobari, G. Funning, E. Papalexakis, S. Oymak, P. Brisk, and E. Keogh, "Matrix profile XVIII: time series mining in the face of fast moving streams using a learned approximate matrix profile," in *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 936–945.

[9] C.-C. M. Yeh, N. Kavantzas, and E. Keogh, "Matrix profile VI: Meaningful multidimensional motif discovery," in *2017 IEEE international conference on data mining (ICDM)*. IEEE, 2017, pp. 565–574.

[10] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2002, pp. 693–703.

[11] K. L. Clarkson and D. P. Woodruff, "Low-rank approximation and regression in input sparsity time," *Journal of the ACM (JACM)*, vol. 63, no. 6, pp. 1–45, 2017.

[12] C.-C. M. Yeh, M. Gu, Y. Zheng, H. Chen, J. Ebrahimi, Z. Zhuang, J. Wang, L. Wang, and W. Zhang, "Embedding compression with hashing for efficient representation learning in large-scale graph," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 4391–4401.

[13] R. Dorfman, "The detection of defective members of large populations," *The Annals of mathematical statistics*, vol. 14, no. 4, pp. 436–440, 1943.

[14] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[15] D. P. Woodruff *et al.*, "Sketching as a tool for numerical linear algebra," *Foundations and Trends® in Theoretical Computer Science*, vol. 10, no. 1–2, pp. 1–157, 2014.

[16] M. Ghashami, E. Liberty, J. M. Phillips, and D. P. Woodruff, "Frequent directions: Simple and deterministic matrix sketching," *SIAM Journal on Computing*, vol. 45, no. 5, pp. 1762–1792, 2016.

[17] A. Munteanu, C. Schwiegelshohn, C. Sohler, and D. Woodruff, "On coresets for logistic regression," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[18] Z. Karnin and E. Liberty, "Discrepancy, coresets, and sketches in machine learning," in *Conference on Learning Theory*. PMLR, 2019, pp. 1975–1993.

[19] M. Matheny, R. Singh, L. Zhang, K. Wang, and J. M. Phillips, "Scalable spatial scan statistics through sampling," in *Proceedings of the 24th ACM SIGSPATIAL international conference on advances in geographic information systems*, 2016, pp. 1–10.
[20] D. Minnen, C. Isbell, I. Essa, and T. Starner, "Detecting subdimensional motifs: An efficient algorithm for generalized multivariate pattern discovery," in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE, 2007, pp. 601–606.
[21] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks," in *International conference on artificial neural networks*. Springer, 2019, pp. 703–716.
[22] Y. Gao and J. Lin, "Discovering subdimensional motifs of different lengths in large-scale multivariate time series," in *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 220–229.
[23] S.-Y. Shih, F.-K. Sun, and H.-y. Lee, "Temporal pattern attention for multivariate time series forecasting," *Machine Learning*, vol. 108, pp. 1421–1441, 2019.
[24] C.-C. M. Yeh, Z. Zhuang, J. Wang, Y. Zheng, J. Ebrahimi, R. Mercer, L. Wang, and W. Zhang, "Online multi-horizon transaction metric estimation with multi-modal learning in payment networks," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 4331–4340.
[25] S. D. Anton, L. Ahrens, D. Fraunholz, and H. D. Schotten, "Time is of the essence: Machine learning-based intrusion detection in industrial time series data," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2018, pp. 1–6.
[26] A. Daigavane, K. L. Wagstaff, G. Doran, C. Cochrane, C. Jackman, and A. Rymer, "Detection of environment transitions in time series data for responsive science," *MileTS*, vol. 20, p. 6th, 2020.
[27] T. Nakamura, M. Imamura, R. Mercer, and E. Keogh, "Merlin: Parameter-free discovery of arbitrary length anomalies in massive time series archives," in *2020 IEEE international conference on data mining (ICDM)*. IEEE, 2020, pp. 1190–1195.
[28] C.-C. M. Yeh, Y. Zheng, J. Wang, H. Chen, Z. Zhuang, W. Zhang, and E. Keogh, "Error-bounded approximate time series joins using compact dictionary representations of time series," in *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*. SIAM, 2022, pp. 181–189.
[29] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and information Systems*, vol. 3, pp. 263–286, 2001.
[30] K.-P. Chan and A. W.-C. Fu, "Efficient time series matching by wavelets," in *Proceedings 15th International Conference on Data Engineering (Cat. No. 99CB36337)*. IEEE, 1999, pp. 126–133.
[31] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," *ACM Sigmod Record*, vol. 23, no. 2, pp. 419–429, 1994.
[32] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *Data Mining and Knowledge Discovery*, vol. 26, pp. 275–309, 2013.
[33] "Supplementary site," 2023, https://sites.google.com/view/sketchdiscord.
[34] S. Huang, H.-K. Wang, and W. Holzer, "3 critically injured in stabbing on taipei metro (update)," *Central News Agency*, 2014-05-21.
[35] Taipei City Government, "data.taipei," 2017, https://data.taipei/.
[36] C.-C. M. Yeh, Y. Zhu, H. A. Dau, A. Darvishzadeh, M. Noskov, and E. Keogh, "Online amnestic dtw to allow real-time golden batch monitoring," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2604–2612.
[37] Wikipedia contributors, "Nangang c3 field — Wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Nangang_C3_Field&oldid=995616850, 2020, [Online; accessed 16-February-2022].
[38] ——, "Mayday (taiwanese band) — Wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Mayday_(Taiwanese_band)&oldid=1071431609, 2020, [Online; accessed 16-February-2022].
[39] C. M. Ahmed, V. R. Palleti, and A. P. Mathur, "WADI: a water distribution testbed for research in the design of secure cyber physical systems," in *Proceedings of the 3rd international workshop on cyber-physical systems for smart water networks*, 2017, pp. 25–28.
[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.
[41] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.
[42] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.
[43] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, pp. 1–27, 2011.
[44] C.-C. M. Yeh, H. Chen, Y. Fan, X. Dai, Y. Zheng, V. Lai, J. Wang, Z. Zhuang, L. Wang, W. Zhang, and E. K. Keogh, "Ego-network transformer for subsequence classification in time series data," in *2023 IEEE International Conference on Big Data (Big Data)*. IEEE, 2023.
[45] A. Der, C.-C. M. Yeh, Y. Zheng, J. Wang, H. Chen, Z. Zhuang, L. Wang, W. Zhang, and E. Keogh, "Time series synthesis using the matrix profile for anonymization," in *2023 IEEE International Conference on Big Data (Big Data)*. IEEE, 2023.

APPENDIX

In this section we provide an analysis of why our algorithm provides high probability of recovery. We borrow from the analysis of the classic Count-Sketch [10] for finding heavy hitters among data streams from a domain size $[d]$.

a) Modeling of Algorithm.: As mentioned above, we say h is drawn from a 2-universal family $\mathcal{H}$ of hashing functions, that is, so that for any two distinct dimensions $j, j' \in [d]$ that $\mathsf{Pr}_{h\sim\mathcal{H}}[h(j) = h(j')] = 1/k$.

We now argue that when we estimate the value $T_i^{(j)}$ of the jth time series at a time point i. In particular, we use $s(j)R_i^{(g)}$ where $j \in J_g$ so $g = h(j)$. Following the standard analysis of the Count-Sketch [10] we obtain the following.

Lemma 1. For $g = h(j)$ then for any time point i we have an unbiased estimate $E[s(j)R_i^{(g)}] = T_i^{(j)}$
and its variance is $Var[s(j)R_i^{(g)}] = \frac{1}{k}\sum_{j'\neq j} T_i^{(j')}$.

Proof. Let $Y_{j',g} = 1$ if $h(j') = g$ and 0 otherwise. The expected value of the sketched time series $s(j)R_i^{(g)}$ for $g = h(j)$ can be factored as follows:

$$E[s(j)R_i^{(g)}] = s(j)s(j)T_i^{(j)} + \sum_{j'\neq j} E[Y_{j',g}s(j)s(j')T_i^{(j')}].$$

Since s is pairwise independent and independent of h

$$\begin{aligned} E[Y_{j',g}s(j)s(j')T_i^{(j')}] &= E[Y_{j',g}s(j)]E[s(j')]T_i^{(j')} \\ &= E[Y_{j',g}s(j)] \cdot 0 \cdot T_i^{(j')} = 0. \end{aligned}$$

All that remains is $s(j)s(j)T_i^{(j)} = T_i^{(j)}$, as desired.

To bound the variance, using our expectation bound, we calculate

$$\begin{aligned} Var[s(j)R_i^{(g)}] &= E[(s(j)R_i^{(g)} - E[s(j)R_i^{(g)}])^2] \\ &= E[(\sum_{j'\neq j} s(j)s(j')Y_{j',g}T_i^{(j')})^2] \\ &= E[\sum_{j'\neq j}\sum_{j''\neq j} s(j'')s(j')Y_{j',g}Y_{j'',g}T_i^{(j')}T_i^{(j'')}] \\ &= \sum_{j'\neq j}\sum_{j''\neq j} T_i^{(j')}T_i^{(j'')}E[s(j'')s(j')Y_{j',g}Y_{j'',g}] \end{aligned}$$

Since s is pairwise independent, $E[s(j')s(j'')] = E[s(j')]E[s(j'')] = 0$ for $j' \neq j''$. Hence all that remains from the variance are terms when $j' = j''$ as

$$Var[s(j)R_i^{(g)}] = \sum_{j' \neq j} (T_i^{(j')})^2 E[Y_{j',g}^2] = \sum_{j' \neq j} (T_i^{(j')})^2 E[Y_{j',g}]$$
$$= \frac{1}{k} \sum_{j' \neq j} (T_i^{(j')})^2. \qquad \square$$

Now recall that we have z-normalized every time series, so the expected value of $(T_i^{(j')})^2$ (over the choice of t) is 1. Thus

$$Var[s[j]R_i^{(g)}] = \frac{1}{k} \sum_{j' \neq j} (T_i^{(j')})^2 = \frac{d-1}{k}.$$

b) Analysis of Subsequences with Large Discord Scores.: Next consider that time series $T^{(j)}$ has a discord with large discord scores, we can now show it is likely to appear as a discord in $R^{(g)}$ – the sketched time series of $\mathbf{T}$, where $g = h(j)$. Considering the time series with subsequences of length m, for a subsequence $T_{i,m}$ in the test set, if the closest subsequence in the training set $T_{i',m}^{(j)}$ is a perfect match (e.g., $T_{i,m}^{(j)} = T_{i',m}^{(j)}$), then we have that for the corresponding spots in the associated sketched time series $R^{(g)}$ that $E[s(j)R_{i,m}^{(g)}] = T_{i,m}^{(j)} = T_{i',m}^{(j)} = E[s(j)R_{i',m}^{(g)}]$. On the other hand, consider a discord that has large discord score, so $T_{i,m} - T_{i',m} = \Delta \in \mathbb{R}^m$, and $\|\Delta\|$ is large. In this case

$$E[s(j)R_{i,m}^{(g)}] = T_{i,m}^{(j)} = T_{i',m}^{(j)} + \Delta = E[s(j)R_{i',m}^{(g)}] + \Delta.$$

So in expectation, the large discord score Δ, measured on the aggregated times series is in expectation still Δ.

We want to understand how large does $\|\Delta\|$ need to be to show up against the noise inherent in the sketched aggregation step. To do this, we can calculate the variance of a subsequence. In particular we have

$$Var[R_{i,m}^{(g)}] = Var[\sum_{t \in [i,\ldots,i+m]} R_t^{(g)}] = m^2 Var[R_{i,m}^{(g)}] = m^2 \frac{d-1}{k}.$$

So if $\|\Delta\|^2 = \|R_{i,m}^{(g)} - R_{i',m}^{(g)}\|^2 = dist(R_{i,m}^{(g)}, R_{i',m}^{(g)})^2$ is much larger than the variance, it will be consistently detected. We can formalize this with a Chebyshev bound.

$$Pr[dist(R_{i,m}^{(g)}, R_{i',m}^{(g)})^2 \geq \alpha] \leq \frac{Var[R_{i,m}^{(g)}]}{\alpha^2} = \frac{m^2 \frac{d-1}{k}}{\alpha^2}.$$

If we set $k = \sqrt{d}$ (as we do in the implementation), and $\alpha = \frac{1}{\sqrt{\delta}} m d^{1/4}$, then we have for any $\delta \in (0,1)$ that

$$Pr[dist(R_{i,m}^{(g)}, R_{i',m}^{(g)}) \geq \frac{1}{\sqrt{\delta}} m d^{1/4}] \leq \delta.$$

Thus if the discord is sufficiently large, it will be detected with high (at least $1-\delta$) probability. Notably, this is with regard to the randomness in the selection of the hash functions (placement in group, and the {-1,+1} choice), and not randomness in the data. So this bound is assumption free and can handle adversarial data.

c) Modeling of Non-adversarial Data.: We can refine this analysis to show high probability of recovering a large discord by considering non-adversarial modeling of the data. Without these assumptions, adversarial unlikely settings may occur. For instance, if each test data subsequence has exactly one similar subsequence in the training data (standard discord analysis will show small discord scores, but it is not robust). Moreover, these matches could be misaligned temporally across dimensions, so the sketched time series presents these non-robust matches as significant discords. If many such cases occur, this can causes a large difference between the discord found by searching sketched time series (as in our algorithm) and from searching the time series for discords individually.

Specifically we now consider an *η-periodic assumption*: that is, that each time series $T^{(j)}$ has a period p_j (it has a rough pattern that repeats after p_j time steps), and for any two time series $T^{(j)}$ and $T^{(j')}$ that p_j and $p_{j'}$ have a small common factor. That is, there exist small positive integers a_j and $a_{j'}$ so that $P = p_j a_j = p_{j'} a_{j'}$. This ensures that all time series have some common latent structure which controls their frequency, and every period P they re-align. For example, periods could be daily, weekly, or hourly and moreover common structural shifts (e.g., day-light-savings-time) affect the latent structure but keeps the general periodic alignment in place. This assumption ensures that for each time window in the test data, that for each period P in the test data there exists a corresponding time window in the training data that has a similar phase across *all* individual time series. Within each alignment there may of course be small fluctuations, and noise, but not structural ones – unless there is a true large discord. In particular, we assume after z-normalizing, that each $T_i^{(j)}$ has variance $Var[T_i^{(j)}] \leq \eta^2$. So if $T_i^{(j)}$ in the test aligns to its corresponding point $T_{i'}^{(j)}$ in the training, their difference has expected value $E[T_i^{(j)} - T_{i'}^{(j)}] = 0$ and variance $Var[T_i^{(j)} - T_{i'}^{(j)}] \leq 2\eta^2$. This is interesting when η is significantly smaller than 1. And we assume there are many periods of length P within the span of the training data, and thus the difference in the number of periods for an individual time series of length p_j will not be too different from the number of periods of length P.

Now analyzing the variance under the periodic assumption, we show with high probability that if there is a large discord score, the sketched time series preserves it.

Lemma 2. Under the η-periodic assumption with joint period P, then if there is a single subsequence $T_{i,m}^{(j)}$ of size m, with most similar subsequence in the training data as $T_{i',m}^{(j)}$ and $dist(T_{i,m}^{(j)}, T_{i',m}^{(j)}) > 2\eta m$, then with high probability (at least $1 - d n_{\text{test}} / 2^{n_{\text{train}}/P}$), that subsequence will be detected.

Proof. Via Lemma 1 we have $Var[s(j)R_i^{(g)}] \leq \frac{1}{k} \sum_{j' \neq j} T_i^{(j')}$, where $g = h(j)$. Now on a subsequence from the jth time series, $T_{i,m}^{(j)}$ in the test data, and its periodic match in the training data $T_{i',m}^{(j)}$, we can analyze their difference $\Delta = T_{i,m}^{(j)} - T_{i',m}^{(j)}$. By the η-periodic assumption

that assumes for aligning elements $E[T_{i,m}^{(j)}(t) - T_{i',m}^{(j)}(t)] = 0$ then and $E[\Delta(t)] = 0$ for each index t. Moreover, using that $Var[T_{i,m}^{(j)}(t)] \leq \eta^2$ we have

$$\begin{aligned}
E[\|\Delta\|^2] &= \sum_{t=1}^{m} E[(T_{i,m}^{(j)}(t) - T_{i',m}^{(j)}(t))^2] \\
&= \sum_{t=1}^{m} E[(T_{i,m}^{(j)}(t) - T_{i',m}^{(j)}(t))^2 - E[T_{i,m}^{(j)}(t) - T_{i',m}^{(j)}(t)]^2] \\
&= \sum_{t=1}^{m} Var[(T_{i,m}^{(j)}(t) - T_{i',m}^{(j)}(t))] \\
&\leq \sum_{t=1}^{m} Var[T_{i,m}^{(j)}(t)] + Var[T_{i',m}^{(j)}(t)] \leq 2m\eta^2.
\end{aligned}$$

Now we bound the variance of $\|\Delta\|^2$ assuming that noise on each time point is independent.

$$\begin{aligned}
Var[\|\Delta\|^2] &= E[\|\Delta\|^4] - E[\|\Delta\|^2]^2 \\
&\leq E[\|\Delta\|^4] = \sum_{t,t' \in [1\ldots m]} E[\Delta(t)^2 \Delta(t')^2] \\
&= \sum_{t,t' \in [1\ldots m]} E[\Delta(t)^2] E[\Delta(t')^2] \\
&= \sum_{t,t' \in [1\ldots m]} (2m\eta^2)(2m\eta^2) = 4m^4\eta^4
\end{aligned}$$

Recall that a Chebyshev inequality for a random variable X states that $Pr[|X - E[X]| \geq \alpha] \leq Var[X]/\alpha^2$ for any $\alpha > 0$. Now assume there is a discord with large discord scores that has $dist(T_{i,m}^{(j)}, T_{i',m}^{(j)}) = \tau$, we can bound the probability at another in-period sequence pair $T_{q,m}^{(j)}, T_{q',m}^{(j)}$ with $\Delta' = T_{q,m}^{(j)} - T_{q',m}^{(j)}$ exceeds that distance with a Chebyshev bound

$$\begin{aligned}
Pr[dist(T_{q,m}^{(j)}, T_{q',m}^{(j)}) > \tau] &\leq Pr[|\|\Delta'\|^2 - E[\|\Delta'\|^2]| > \tau^2 - E[\|\Delta'\|^2]] \\
&\leq \frac{Var[\|\Delta'\|^2]}{(\tau^2 - E[\|\Delta'\|^2])^2}.
\end{aligned}$$

If we set $\tau^2 = 4m\eta^2(1 + m/\sqrt{\delta})$ or more simply $\tau > 2m\eta\delta^{-1/4}$, then with some algebra we can achieve

$$Pr[dist(T_{q,m}^{(j)}, T_{q',m}) > \tau] \leq \delta.$$

This is for a simple potential match. If there are $n' = n_{\text{train}}/P$ potential matches (under the periodic assumption), then the distance must exceed this for all n' matches. Let $\|\Delta'_*\| = \min_{\ell \in 1\ldots n'} dist(T_{q,m}^{(j)}, T_{q',m}^{(j)})$. By a union bound, we have $\|\Delta'_*\| > \tau$ only if no n' potential matches exceeds τ which happens with probability at most $\delta^{n'}$ if their noise is independent. Setting $\delta = 1/2$, then if $\tau > (2^{3/4})m\eta > 2m\eta$ then $Pr[\|\Delta'_*\| \geq \tau] \leq 1/2^{n'}$.

Next we need to consider the dn_{test} different subsequences from the d time series which may show up as false discords. Within a time series, these overlap, so are not independent. But we can use a union bound to address the concern than if any of these has more than τ distance from its closest match in the training set it will show up as the discord. If we set $\delta' = (1/2^{n'})dn_{\text{test}}$, and have $\tau > 2m\eta$, then with probability at least $1 - (1/2^{n'} \cdot d \cdot n_{\text{test}})$, if an discord has $\|\Delta\| > \tau$ it will show up as the discord. $\square$